A Multiresolution Approach for Motion Planning Under Differential Constraints

Stephen R. Lindemann and Steven M. LaValle
Department of Computer Science
University of Illinois
Urbana, IL 61801 USA
{slindema, lavalle}@uiuc.edu

***Abstract*— In this paper, we present an incremental, multiresolution motion planning algorithm designed for systems with differential constraints. Planning for these sytems is more difficult than ordinary path planning due to the presence of momentum (drift) or nonholonomic velocity constraints. Given a motion planning problem for such a system and that a solution to the problem exists, then a finite reachability graph containing a solution trajectory is guaranteed to exist, under very reasonable conditions. In general, this graph can be generated using sufficiently dense input space sampling, sufficiently small time step, and sufficiently large tree depth. We show how to find and search such a tree in an incremental, multiresolution way. We prove the completeness of our algorithm, discuss related practical concerns, and show experimental results for several systems.**

I. Introduction

Motion planning algorithms utilizing the configuration space [1] have been developed with great success over the past three decades. However, most motion planning algorithms–including modern sampling-based algorithms–are designed and used for standard path planning applications not involving differential constraints. Even though these problems are extremely difficult (in fact, PSPACE-hard [2]), planning for systems with differential constraints is much more challenging. For example, the system may have significant momentum yet small control influence, as in the case of a space ship using small thrusters, or the system may not be fully actuated, as in the case of nonholonomic mobile robots. These additional restrictions make it more difficult for a motion planning algorithm to find a feasible trajectory for the system; i.e., one which takes into account both the differential constraints and the global obstacle constraints.

Whereas path planning is more common, planning under differential constraints has been addressed since Laumond introduced nonholonomic planning to the community [3]. Some approaches to planning under differential constraints are based on decomposing the problem into two parts: first, planning a path for the system without considering the constaints; then, transforming the path to satisfy the constraints [4]–[8]. These planners have the same completeness properties as the underlying path planner. Sampling-based approaches using random sampling are typically *probabilistically complete*: that is, the probability of finding a feasible path approaches one as the number of iterations goes to infinity, assuming that a solution path exists. Examples of such planners include [9]–[11]. Barraquand and Latombe [12] gave the first *resolution complete* planner for nonholonomic multibody vehicles. Resolution complete planners are guaranteed to find a solution trajectory in finite time if one exists, assuming the resolution of the search is fine enough. Cheng and LaValle introduced resolution complete RRTs [13]. Resolution completeness is advantageous compared to probabilistic completeness because it offers a strong deterministic performance guarantee, rather than probabilistic bounds. Our algorithm extends Barraquand and Latombe's work and consequently is resolution complete. We begin with a large search resolution and refine it incrementally in a multi-resolution way. Figure 1 illustrates the type of refinement our planner produces for a car-like robot.

Section II, briefly defines the problem of planning under differential constraints and describes related work. Section III discusses Barraquand and Latombe's planner in more detail, and presents our algorithm and its proof of resolution completeness. Practical considerations and demonstrations of our algorithm are given in Section V.

II. Problem Formulation and Related Work

This section formally defines the problem we consider and discusses related work for planning under differential constraints.

A. Problem Formulation

The basic motion planning problem is to find a continous path in the configuration space from an initial state to a goal state, both of which are given. In the problem of motion planning under differential constraints, planning is performed in a state space (often the phase space of the configuration space) and the trajectory must satisfy the differential constraints at every point. There can be both equality and inequality constraints; the equality constraints are often expressed in the state transition equation, $\dot{x} = f(x, u)$. A solution trajectory is constructed by computing a function $u : [0, t_f] \rightarrow U$ such that the final position $x(t) = x(0) + \int_0^{t_f} f(x(s), u(s))ds$ is the goal state, x_{goal}. In general, the input set is a bounded subset of $\mathbb{R}^m$. For the present, assume that we are given a finite input set, U. Later, we will see that this is not necessary; however, it will facilitate the description of our algorithm. Our algorithm will return a finite sequence of inputs, each having

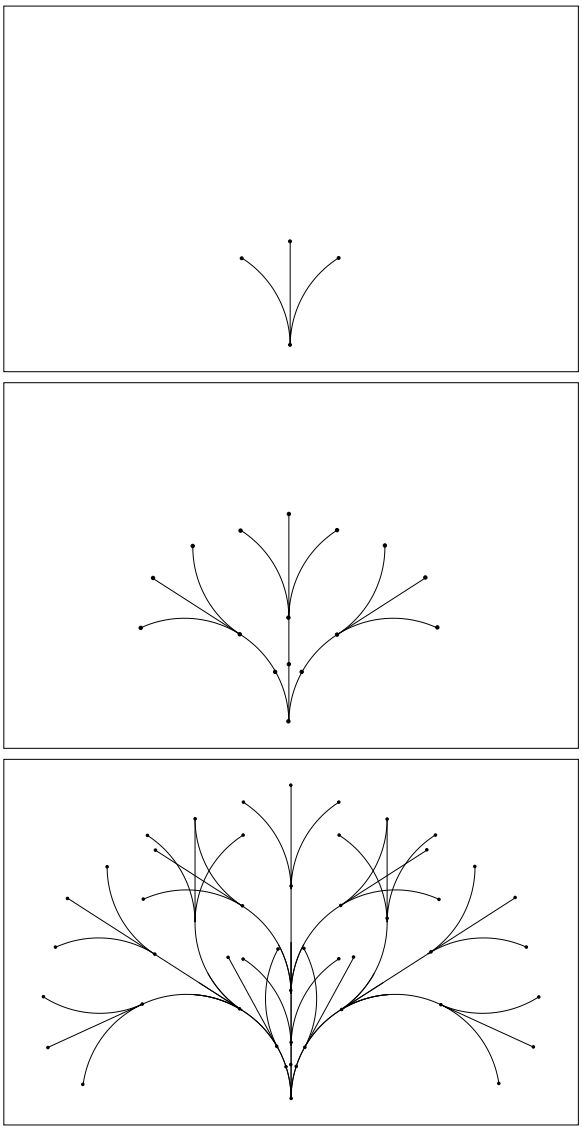

Fig. 1. From top to bottom: search graphs consisting of nodes of ranks 1, 2, and 3.

finite application time, which takes the system from the initial state to the goal state.

B. Planning Under Differential Constraints

As discussed in Section I, planning under differential constraints takes several forms. We will not discuss "plan and transform" planners, as they are not as relevant to our work. Instead, we focus on several probabilistically complete or resolution complete algorithms which compute solution trajectories directly. We delay discussing Barraquand and Latombe's algorithm until Section III.

Hsu *et al.* introduced an expansive-space planning algorithm for kinodynamic robots and moving obstacles [9]; the algorithm was later called the Expansive Space Tree (EST) method. In this algorithm, a node in the tree is chosen according to a biased probability measure, and an input from the control set is chosen at random and applied from that node. If the resulting trajectory is collision-free, then the new node is added to the tree. In order to take moving obstacles into account, sampling and collision checking was done in the state$\times$time space.

LaValle and Kuffner developed Rapidly-Exploring Random Trees (RRTs) for motion planning under differential constraints [10]. RRTs select a state at random from the state space and find its nearest neighbor in the search graph. Then, an input is applied which advances toward the random state. The sampling and expansion process is Voronoi-biased and tends to grow the search tree toward unexplored areas quickly.

Both of the above planners are probabilistically complete; Cheng and LaValle developed an RRT variant which is resolution complete [13]. It achieves this completeness by discretizing the state space and ensuring exhaustive search of the reachability graph.

Frazzoli *et al.* introduced the idea of maneuver-based motion planning for nonlinear systems with symmetries [14], [15]. Some of their work is based on integrating maneuvers (or *motion primitives*) with RRT-based search. In the context of safety verification, Bahtia and Frazzoli introduce an RRT-based method which refines the time discretization, a property which our algorithm shares.

Finally, Ladd and Kavraki presented the PDST-EXPLORE planner [11], designed to avoid some of the difficulties to which other sampling-based algorithms are prone to when applied to systems with differential constraints. This planner chooses a trajectory segment based on a deterministic update schedule and then creates a new segment extending from a point in the existing segment, through the application of a random control. This method is also probabilistically complete.

III. Exploring the Reachability Graph Incrementally and Exhaustively

In this section, we will describe our algorithm and prove that it is resolution complete. Since our method is inspired by the nonholonomic planner of Barraquand and Latombe [12], we will describe their method and show how our planner significantly extends it.

A. Barraquand and Latombe's Planner

In [12], Barraquand and Latombe present a resolution complete algorithm for finding feasible (even optimal) trajectories for nonholonomic mobile multi-body vehicles. Their algorithm is simple yet innovative. It uses three parameters: the time discretization, Δt (they use the notation δt_0); the maximum allowable depth of the search tree, H; and the discretization of the state space, R (the space is decomposed into a grid having 2^R cells per axis). The first two parameters define a search tree for the algorithm; the initial state is the root of the tree, and every child node is formed by applying a fixed control for length Δt from a parent node. Denote this tree as $T(\Delta t, H)$. For the problems they consider, the system is controllable using only four inputs (corresponding to forward/reverse and the extremal steering angles); hence, the total number of nodes in $T(\Delta t, H)$ is 4^H. Nodes are expanded one at a time using a priority queue (they use the number of path reversals as

the priority). The algorithm will not necessarily expand all nodes in $T(\Delta t, H)$, however, because they require that each discrete cell in the state space contain at most one node of the tree. They call their planner *asymptotically complete*, which means that if the parameters are chosen fine enough, a solution will be found by the planner. Their proof of completeness has three steps. First, they assert that if a solution path exists for the continuous time systems they consider, then there exists a small enough value of Δt so that a solution path exists which is generated by a finite sequence of controls, each applied for time Δt. Second, H can be chosen large enough so that this path exists in $T(\Delta t, H)$. Third, the state space discretization R can be chosen fine enough that every node in $T(\Delta t, H)$ is in a separate cell, guaranteeing that the path will be found during the search process, regardless of the order in which the nodes of $T(\Delta t, H)$ are expanded. Note that only Δt and H are needed for completeness, not R. The parameter R serves two other purposes in their planner. The first purpose is as a heuristic; it forces the search to be more uniform over the entire configuration space and potentially reduces the number of nodes expanded. Secondly, it can be used to discretize the obstacles and perform collision checking; however, as they note in their paper, there are better ways to perform this task.

The biggest disadvantage of Barraquand and Latombe's method is that there is no way to know ahead of time what the parameter values should be in order for a solution to be found. If a search fails, it is unclear whether the search depth should be increased, the time step shortened, or the state space resolution increased. Additionally, poor choices of time and state space discretization can lead to problems. If the time discretization is so fine that the system cannot move from one cell to another in a single time interval, then no progress will be made. This means that the state space discretization must be made similarly fine, which potentially decreases its heuristic value. We address these problems by describing a planner which incrementally refines the search tree in a multiresolution way.

B. Proposed Method

Our method can be seen as an multiresolution extension of Barraquand and Latombe's planner. Our method incrementally decreases the time discretization, increases the depth of the search, and refines the state space discretization. The proof of completeness of our planner, like that of Barraquand and Latombe, is structured around two key statements: first, with Δt small enough and H large enough, a solution trajectory will exist in $T(\Delta t, H)$; and second, that this tree will be generated and searched exhaustively given a fine enough state space discretization. We assume that the first statement is true; there are certain requirements on the system for this to be the case, but a detailed discussion of issues related to this question is beyond the scope of this work. Significantly more detail can be found in [16]. Our algorithm searches an infinite sequence of increasingly higher resolution reachability graphs, until either the search is stopped or a solution trajectory is found. If a solution trajectory exists, it is guaranteed to be found by the planner.

```
GENERATE_SEARCH_TREE(initialState)
1   createNewNode(initialState)
2   while (!solutionFound &&
            iterations < maxIterations)
3       x = chooseNode()
4       if (isTrajectoryValid(parent(x), x))
5           T.insert(x)
6           for u in U
7               createNewNode(Child(x, u, 0))
        // assume x = Child(y, u, i)
8       createNewNode(Child(y, u, i + 1))
9       iterations = iterations + 1
10  end while
```

Fig. 2. Our algorithm, which generates and searches a multiresolution reachability graph.

Our algorithm expands nodes in the search tree based on the ranks of nodes in the search tree; the rank of a node encodes information regarding both the depth of the node in the tree and the fineness of the time discretization required to reach the node. For the moment, disregard the existence of any state space discretization; as in the case of Barraquand and Latombe, this is not essential for completeness and will be considered later. The two parameters needed for our algorithm are Δt and a "lookahead" parameter l, which we will discuss later. This parameter Δt, in contrast to that of Barraquand and Latombe, should be chosen to be large; it represents the largest step the planner will attempt to take. The planner will reduce the search time increment, as needed. This is related to the R4T method of Bahtia and Frazzoli [17]. Assume that the system transition equation is $\dot{x} = f(x, u)$ and that the available controls are consist of a finite set U. Let $t = 2^{-i}\Delta t, i \in \{0, 1, \ldots\}$; then every node will be generated from a parent node as follows:

$$Child(x, u, i) = x + \int_0^t f(x, u) d\tau$$

Then the rank of the node is defined to be:

$$Rank(Child(x, u, i)) = Rank(x) + i + 1$$

The rank of the initial state, which is the root of the tree, is taken to be zero. Since i is greater than or equal to zero, the rank of a child is always greater than that of its parent.

Now, we can formally describe our algorithm, which we do in Figure III-B. When a node is chosen for expansion in $chooseNode$, it is added to the search tree if the trajectory connecting it to its parent is collision-free. If this occurs, a number of children are generated, corresponding to the inputs applied for the maximum time length Δt. Regardless of whether or not the node is added to the search tree, a new node is generated from the parent at a smaller time discretization. This means that the tree is continually being refined. When a node is created, it is eligible for selection by $chooseNode$ (i.e., it is placed in a list of open nodes); a node is removed from the open list when it is selected for expansion.

The order in which $chooseNode$ selects nodes is critical for resolution completeness. However, finding sufficient conditions to apply is not difficult. Given some $l \in \mathbb{Z}^+$, assume that $chooseNode$ never expands a node of rank $i + l$ until all nodes of rank i have been expanded. This is sufficient to guarantee resolution completeness, which is formally stated in the following theorem:

Theorem 1 *If $chooseNode$ never expands a node of rank $i+l$ until all nodes of rank i have been expanded, then the planner is resolution complete.*

Proof: We assume that there exists a reachability graph of time discretization $t = 2^{-i}\Delta t$ and depth H, denoted $T(t, H)$, such that a solution path exists in $T(t, H)$. As mentioned above, conditions for this are discussed in [16]. In order to show resolution completeness, we need to guarantee that the algorithm will generate and search this entire tree. First, we show that this tree is generated in its entirety (at least, all collision free paths in the tree exist). We will use induction to show that for any rank r, all nodes generated by collision-free paths having rank r will be generated. Having done this, it will be simple to show that they will be searched.

As our base case, it is obvious that all nodes with rank $r = 1$ are added, since the root node is expanded in the first iteration (since it is the only node in the tree at that time). The children it generates during that iteration consist of the entire set of nodes of rank 1. Now, assume that all nodes of rank $j, j \leq r$ have been generated; we need to show that all nodes of rank $r+1$ will eventually be generated. Note that all nodes of rank $j, j \leq r$ must be chosen for expansion in finite time, since there are finitely many of these nodes and they must all be expanded before any node of rank $r+l$ is expanded. Every node of rank r is generated as a result of the expansion of some node with rank less than r; this can be seen since the new nodes generated in lines 7 and 8 of the algorithm all have rank equal to one greater than the rank of the node chosen for expansion. Hence, every node of rank r will be generated in finite time.

Now, we show that the tree $T(t, H)$ will be exhaustively searched. The maximum rank of any node in $T(t, H)$ is $(i + 1)H$, since the maximum depth is H and the rank grows by at most $i + 1$ for any node (since we have $t = 2^{-i}\Delta t$ for this tree). By assumption, $chooseNode$ will expand all these nodes before any nodes of rank $(i + 1)H + l$ are expanded. Since the number of nodes of any given rank r is finite, (denote this by N_r), the sum $\sum_{r=1}^{(i+1)H+l-1} N_r$ is also finite and the tree is guaranteed to be searched exhaustively in finite time. Hence, the solution trajectory will be found. ■

Apart from the stated condition, there is great freedom in how $chooseNode$ is permitted to select nodes for expansion. It may be possible to develop selection methods tailored to dynamical systems, or any of a number of standard motion planning heuristics could be applied. For example, the nodes could be selected according to an RRT heuristic [10], so as to reduce dispersion [18], or to maximize information gain [19], [20]. One could also use an optimality criterion to add nodes, as in [12].

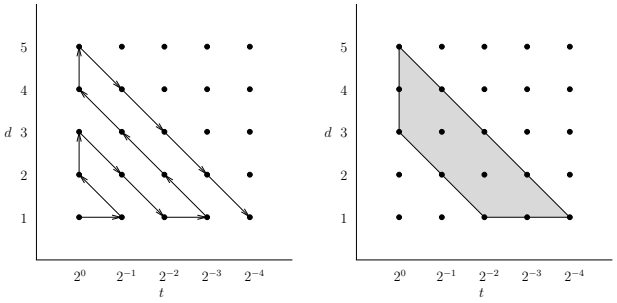

Fig. 3. Different approaches to guaranteeing resolution completeness. On the left, nodes must be added in the order specified; on the right, any node from the strip may be added, and the strip is advanced when all nodes on the lower boundary have been added. Our method is analogous to, though not identical with, this method.

There are a number of different requirements that, if placed on $chooseNode$, will yield resolution completeness. A straightforward approach, given in [21], is to use a "triangularization" approach to adding nodes of particular time discretizations and depths. Our approach is advantageous compared to this one because it provides much more flexibility with respect to the order in which different nodes may be added, without sacrificing the completeness guarantees. An illustration of the differences may be seen in Figure 3.

Our method extends easily to the case where the input set is not finite. Assume that the input set is bounded and that S is a sample sequence that is dense in U, and let S_n be the first n elements of S (alternatively, it could denote the first n "resolution levels" of the sample sequence, which might be on the order of 2^n samples). Let $T(t, H, S_n)$ be the tree generated by applying inputs from S_n, at time discretization t, up to depth H. We can then use a triangularization argument as in [21], or a revised rank criterion, to maintain resolution completeness in this setting. Cheng and LaValle use a sampled input space in their formulation of the resolution completeness problem [16].

Up to this point, we have ignored the issue of state space discretization. While it appeared in Barraquand and Latombe's planner, we argued above that it did not play an important role in the resolution completeness of the planner. Rather, it can be used to improve the efficiency of the planner by forcing the exploration to be more global, while not preventing resolution completeness. Also in the interest of promoting efficient search, we propose an incremental, multi-resolution analogue to Barraquand and Latombe's state space discretization. First, it can be noted that the key use of the discretization was that the tree was not permitted to add a node in a cell that already contained one. This forces new nodes to be added to unexplored cells. However, there is no need to require that the cells in the partition be grid cells, which is what arise from a standard discretization of the space. Any kind of partition will work for this purpose, although some partitions will certainly be better than others [21]. We propose that the

Voronoi diagram of a finite point set P be used to partition the space. Thus, a node with nearest neighbor $p \in P$ would not be added if another node in the tree exists with p as its nearest neighbor.

This partition can be refined in an incremental way by adding more points to the point set P. In order for the partition to be useful, the point sequence generating P needs to be dense, uniform, and incremental; possible choices of sample sequences are random or quasi-random points or incremental grid sequences [22]. Note that the partition is only refined in a loose sense; technically, it is not a refinement because a post-refinement cell may not be contained entirely in any pre-refinement cell, although performance might be improved if this was the case. It should be noted that after a refinement, the tree may have two or more nodes present in a single cell; this is acceptable, but we continue to require that no new nodes can be added in that cell if this occurs.

Assuming we utilize a state space partitioning scheme as above, we need to show that the resolution completeness of our planner is not detrimentally affected. First, define the set of *open nodes* to be all nodes eligible for expansion without considering the state space partition; for some i, they will all have some rank r satisfying $i \leq r < i + l$. We have already shown that this set will always have size greater than zero. Define the set of *valid nodes* to be those open nodes which are eligible for expansion under the state space partition. We now permit $chooseNode$ to select for expansion only valid nodes, rather than simply open nodes. Then, assume that the sample sequence generating P is dense. Then, we have the following:

Theorem 2 *Let $chooseNodes$ be as in Theorem 1, together with the requirements given above. If the partition is refined any time the number of valid nodes is zero, then the algorithm is resolution complete.*

Proof: The number of open nodes is guaranteed never to reach zero. Since the sample sequence is dense, there exists some number of samples n such that if $P = \{s_1, s_2, \cdots, s_n\}$, each open node is in a separate cell of the partition. Then every open node is also valid, and will be chosen by $chooseNode$. Thus, the resolution completeness guarantee still holds. ■

IV. Practical Considerations and Experiments

In our implementation of this planner, we encountered several practical issues which deserve to be addressed. We believe that they are related to fundamental issues in sampling-based planning under differential constraints and give direction for future research.

Motion Primitives: The value of motion primitives, introduced by Frazzoli *et al.*, has become more apparent in recent years, both in the context of RRTs (as in [14]), and in other planners such as PDST-Explore [11]. In the context of our planner, carefully-designed motion primitives offer great advantages over simply applying constant inputs. Recall that we set the initial Δt quite large, with the goal of permitting the planner to take large steps in a single iteration; however, applying a constant input over a long time interval is often undesirable for systems with significant dynamics. For example, applying the same input may give the system far too much momentum in one direction. It is far more useful to apply an input for a shorter length of time and utilize the rest of the time to coast, or to apply inputs which balance each other, thus keeping the system momentum at more manageable levels. If motion primitives can be designed which can be applied for any Δt, then they fit perfectly with our planner; ideally, we would like to have a range of inputs which scale from carefully-designed motion primitives at large Δt to appropriate constant inputs for small Δt. We believe that finding good sets of motion primitives is a key issue in planning for systems with differential constraints.

Time Discretization and State Space Partitioning: A choice of appropriately large time discretization is not only advantageous insofar as it enables large steps to be taken, but it is also important when using the state space partitioning method we describe. The motivation for using our method as opposed to a strict grid discretization as in [12] is that the partition can be initialized with only a few points and then can be refined incrementally as needed. Our planner is still affected by the problem described earlier for Barraquand and Latombe's planner: if the time discretization is fine (i.e., Δt is too small), then a fine state space partition is needed; otherwise, no nodes can be added, since only one tree vertex per cell is permitted.

Lookahead and Heuristic Search: The choice of the lookahead parameter has significant impact on the way that the planner searches the reachability graph. If the lookahead parameter is very large (recall that it can be any finite non-negative integer), then the behavior of the planner will mimic that of the heuristic used to guide $chooseNode$ (after taking into account the state space partition requirement). If the lookahead is zero, the planner essentially performs breadth first search. A small lookahead parameter often interacts with the state space partition in the same way that a small Δt does, reducing the efficiency of the planner for simple problems. This is because a small lookahead causes the planner to add nodes close to the initial state early in the search, and the state space partition must become very fine in order to permit these nodes to be added. For many problems, a reasonably large lookahead is preferable, because it retains resolution completeness while exploiting the (hopefully well-informed) heuristic search behavior in $chooseNode$.

Figure 4 illustrates paths found by our algorithm. In the first experiment, a Reeds-Shepp car must navigate a maze. In this example, motion primitives were not used. As can be seen in the figure, the large Δt causes there to be many reversals. This could be alleviated by introducing motion primitives which regulate the amount the car can turn during one Δt. In the second experiment, we solve a lane changing problem where a car must swerve to avoid the obstacle while traveling at a large velocity. The state space of the vehicle has 5 degrees of freedom, incorporating sliding dynamics into the motion

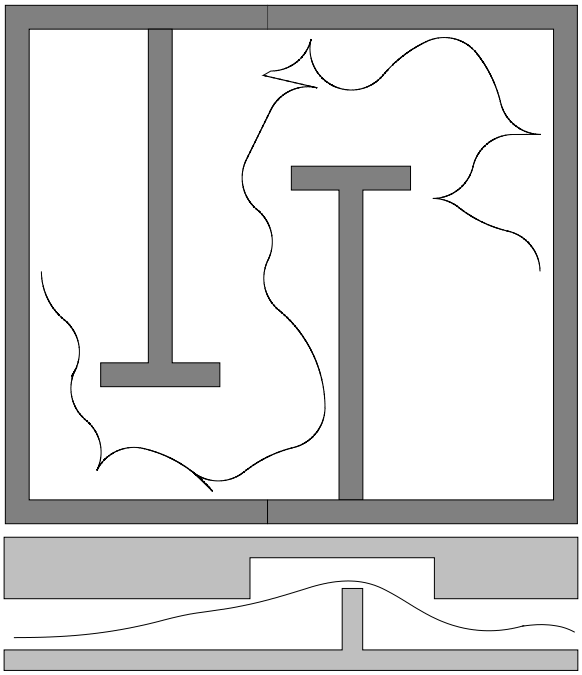

Fig. 4. Two sample trajectories found by our planner. On top, a Reeds-Shepp car must navigate the maze; on bottom, a car with sliding dynamics performs a high-speed lane change.

of the car. We used constant inputs as well as basic motion primitives in the construction of the search tree.

V. Conclusion

In conclusion, we have presented a new motion planning algorithm for systems with differential constraints. Our work is an incremental, multi-resolution extension of Barraquand and Latombe's seminal planner for multibody mobile robots. We proved that our planner is resolution complete, both with and without the use of a state space partition. Our planner provides a platform for providing these strong deterministic guarantees while utilizing popular heuristic search techniques to enhance practical performance. We implemented our planner and have tested it on several systems with differential constraints. We have also identified key practical considerations for our planner. In the future, we plan to address these areas in more depth. We would also like to develop good sets of motion primitives for common dynamical systems and to test our planner on more difficult planning problems.

Acknowledgment

This work was funded in part by NSF Awards 9875304, 0118146, and 0208891.

References

[1] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.

[2] J. H. Reif, "Complexity of the mover's problem and generalizations," in *Proc. of IEEE Symp. on Foundat. of Comp. Sci.*, 1979, pp. 421–427.

[3] J. P. Laumond, "Trajectories for mobile robots with kinematic and environment constraints," in *Proc. of International Conference on Intelligent Autonomous Systems*, 1986, pp. 346–354.

[4] P. Ferbach, "A method of progressive constraints for nonholonomic motion planning," in *IEEE Int. Conf. Robot. & Autom.*, 1996, pp. 2949–2955.

[5] J. P. Laumond, S. Sekhavat, and F. Lamiraux, "Guidelines in nonholonomic motion planning for mobile robots," in *Robot Motion Plannning and Control*, J.-P. Laumond, Ed. Berlin: Springer-Verlag, 1998, pp. 1–53.

[6] S. Sekhavat, P. Svestka, J.-P. Laumond, and M. H. Overmars, "Multilevel path planning for nonholonomic robots using semiholonomic subsystems," *Int. J. Robot. Res.*, vol. 17, pp. 840–857, 1998.

[7] F. Bullo and K. M. Lynch, "Kinematic controllability for decoupled trajectory planning in underactuated mechanical systems," *IEEE Trans. on Robotics and Automation*, vol. 17, no. 4, pp. 402–412, 2001.

[8] P. Choudhury and K. Lynch, "Trajectory planning for second-order underactuated mechanical systems in presence of obstacles," in *Proceedings of the Workshop on Algorithmic Foundations of Robotics*, 2002.

[9] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," in *Algorithmic and Computational Robotics: New Directions*, B. R. Donald, K. M. Lynch, and D. Rus, Eds. Wellesley, MA: A K Peters, 2001.

[10] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," in *Algorithmic and Computational Robotics: New Directions*, B. R. Donald, K. M. Lynch, and D. Rus, Eds. Wellesley, MA: A K Peters, 2001, pp. 293–308.

[11] A. M. Ladd and L. E. Kavraki, "Fast exploration for robots with dynamics," in *Proc. Workshop on Algorithmic Foundation of Robotics*, 2004, pp. 313–328.

[12] J. Barraquand and J.-C. Latombe, "Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles," *Algorithmica*, vol. 10, pp. 121–155, 1993.

[13] P. Cheng and S. M. LaValle, "Resolution complete rapidly-exploring random trees," in *Proc. IEEE Int'l Conf. on Robotics and Automation*, 2002, pp. 267–272.

[14] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *AIAA Journal of Guidance and Control*, vol. 25, no. 1, pp. 116–129, 2002.

[15] ——, "Maneuver-based motion planning for nonlinear systems with symmetries," *IEEE Trans. Robot. & Autom.*, 2005.

[16] P. Cheng and S. M. LaValle, "Resolution completeness for sampling-based motion planning with differential constraints," *Submitted to the International Journal of Robotics Research*, 2004.

[17] A. Bhatia and E. Frazzoli, "Incremental search methods for reachability analysis of continuous and hybrid systems," in *Hybrid Systems: Computation and Control*, Mar. 2004.

[18] S. R. Lindemann and S. M. LaValle, "Incrementally reducing dispersion by increasing Voronoi bias in RRTs," in *Proc. IEEE International Conference on Robotics and Automation*, 2004.

[19] B. Burns and O. Brock, "Information theoretic construction of probabilistic roadmaps," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2003, pp. 650–655.

[20] ——, "Sampling-based motion planning using predictive models," in *IEEE Int. Conf. Robot. & Autom.*, 2005.

[21] S. M. LaValle, *Planning Algorithms*. Cambridge University Press (also available at http://msl.cs.uiuc.edu/planning/), to be published in 2006.

[22] S. R. Lindemann, A. Yershova, and S. M. LaValle, "Incremental grid sampling strategies in robotics," in *Workshop on the Algorithmic Foundations of Robotics*, 2004, pp. 297–312.